

A Gradient-Based Neural Network Method for Solving Strictly Convex Quadratic Programming Problems

Alireza Nazemi · Masoomeh Nazemi

Received: 26 October 2012 / Accepted: 3 February 2014 / Published online: 21 February 2014
© Springer Science+Business Media New York 2014

Abstract In this paper, we study a gradient-based neural network method for solving strictly convex quadratic programming (SCQP) problems. By converting the SCQP problem into a system of ordinary differential equation (ODE), we are able to show that the solution trajectory of this ODE tends to the set of stationary points of the original optimization problem. It is shown that the proposed neural network is stable in the sense of Lyapunov and can converge to an exact optimal solution of the original problem. It is also found that a larger scaling factor leads to a better convergence rate of the trajectory. The simulation results also show that the proposed neural network is feasible and efficient. The simulation results are very attractive.

Keywords Neural network · Convex quadratic programming · Fischer-Burmeister function · Convergent · Stability

Introduction

It is well known that SCQP problems arise in a wide variety of scientific and engineering applications including regression analysis, image and signal processing, parameter estimation, filter design, robot control, etc. [5]. In many

engineering and military applications, the demand on real-time SCQP solutions is often needed, such as classification in complex electromagnetic environments, recognition in medical diagnostics radar object recognition in strong background clutter [16]. Since the computing time required for solving quadratic optimization problem greatly depends on the dimension and the structure of the problem, the conventional numerical methods [1–47] are usually less effective in real-time applications. One promising approach for handling these optimization problems with high dimensions and dense structure is to employ an artificial neural network [31–46] based on circuit implementation. We consider that it is appropriate to utilize the neural networks for efficiently solving the SCQP problems.

The dynamic system approach is one of the important methods for solving optimization problems, which was first proposed by Pyne in the late 1950s [31]. Artificial recurrent neural networks for solving constrained optimization problems can be considered as a tool to transfer the optimization problems into a specific dynamic system of first-order differential equations. It is expected that for an initial point, the dynamic system will approach its static state (or equilibrium point) which corresponds to the solution of the underlying optimization problem. An important requirement is that the energy function decreases monotonically as the dynamic system approaches an equilibrium point. Because of the dynamic nature and the potential of electronic implementation, neural networks can be implemented physically by designated hardware such as application-specific integrated circuits, where the computational procedure is truly distributed and parallel. Therefore, the neural network approach can solve optimization problems in running time orders of magnitude faster than the most popular optimization algorithms executed on general purpose digital computers. It is of great interest in practice to develop some

A. Nazemi (✉)
Department of Mathematics, School of Mathematical Sciences,
Shahrood University, P.O. Box 3619995161-316, Shahrood, Iran
e-mail: nazemi20042003@yahoo.com

M. Nazemi
Department of Food Science and Technology, Damghan Branch,
Islamic Azad University, Damghan, Iran
e-mail: masomehni@yahoo.com

neural network models which could provide a real-time online solution for the SCQP problem.

Motivated by the above discussions, in this paper a gradient neural network model based on Fischer-Burmeister (FB) function for solving the SCQP is presented. According to the saddle point theorem, the equilibrium point of the proposed neural network is proved to be equivalent to the Karush Kuhn Tucker (KKT) point of the SCQP problem. The existence and uniqueness of an equilibrium point of the proposed neural network are analyzed. By constructing a suitable Lyapunov function, a sufficient condition to ensure globally stable in the sense of Lyapunov for the unique equilibrium point of the neural network is derived.

The rest of the paper is organized as follows. In “Problem formulation” section, an NCP-function is used to reformulate the SCQP problem as an unconstrained minimization problem with objective energy function. In “A gradient neurodynamic model” section, a gradient neural network is constructed to solve the SCQP problem, and the stability properties of the proposed neural network are investigated. Some simulation results are discussed to evaluate the effectiveness of the proposed neural network in “Computer simulations” section. Finally, section “Conclusion” concludes this paper.

Problem Formulation

Consider the following SCQP problem:

$$\text{minimize } \frac{1}{2}x^T Qx + d^T x \tag{1}$$

subject to

$$Ax - b \leq 0, \tag{2}$$

$$Gx - f = 0, \tag{3}$$

where $Q \in \mathbb{R}^{n \times n}$ is only a symmetric and positive definite matrix, $d \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $G \in \mathbb{R}^{l \times n}$, $f \in \mathbb{R}^l$, $x \in \mathbb{R}^n$ and $\text{rank}(A, G) = m + l$ ($0 \leq m, l < n$).

For the convenience of later discussions, it is necessary to introduce a few notations, definitions, two lemmas and two theorems. In what follows, $\|\cdot\|$ denotes the l^2 -norm of \mathbb{R}^n , T denotes the transpose and $x = (x_1, x_2, \dots, x_n)^T$. If a differentiable function $\mathcal{H} : \mathbb{R}^n \rightarrow \mathbb{R}$, then $\nabla \mathcal{H} \in \mathbb{R}^n$ stands for its gradient. For any differentiable mapping $\mathcal{H} = (\mathcal{H}_1, \dots, \mathcal{H}_m)^T : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $\nabla \mathcal{H} = [\nabla \mathcal{H}_1(x), \dots, \nabla \mathcal{H}_m(x)] \in \mathbb{R}^{n \times m}$, denotes the transposed Jacobian of \mathcal{H} at x .

Definition 2.1 Let $\Omega \subseteq \mathbb{R}^n$ be an open neighborhood of \bar{x} . A continuously differentiable function $\zeta : \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be a Lyapunov function at the state \bar{x} (over the set Ω) for a system $x' = \mathcal{H}(x)$, if

$$\begin{cases} \zeta(\bar{x}) = 0, \zeta(x) > 0, \quad \forall x \in \Omega \setminus \{\bar{x}\}, \\ \frac{d\zeta(x(t))}{dt} = [\nabla \zeta(x(t))]^T \mathcal{H}(x(t)) \leq 0, \quad \forall x \in \Omega. \end{cases}$$

Lemma 2.2 [24]: (a) An isolated equilibrium point x^* of a system $x' = \mathcal{H}(x)$ is Lyapunov stable if there exists a Lyapunov function over some neighborhood Ω^* of x^* .

(b) An isolated equilibrium point x^* of a system $x' = \mathcal{H}(x)$ is asymptotically stable if there is a Lyapunov function over some neighborhood Ω^* of x^* such that $\frac{d\zeta(x(t))}{dt} < 0, \quad \forall x \in \Omega^* \setminus \{x^*\}$.

Definition 2.3 Let $x(t)$ be a solution trajectory of a system $x' = \mathcal{H}(x)$, and let X^* denotes the set of equilibrium points of this equation. The solution trajectory of the system is said to be globally convergent to the set X^* , if $x(t)$ satisfies

$$\lim_{t \rightarrow \infty} \text{dist}(x(t), X^*) = 0,$$

where $\text{dist}(x(t), X^*) = \inf_{y \in X^*} \|x - y\|$. In particular, if the set X^* has only one point x^* , then $\lim_{t \rightarrow \infty} x(t) = x^*$, and the system is said to be globally asymptotically stable at x^* if the system is also stable at x^* in the sense of Lyapunov.

Theorem 2.4 [5] $x \in \mathbb{R}^n$ is an optimal solution of (1)–(3) if and only if there exist $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^l$ such that $(x^T, u^T, v^T)^T$ satisfies the following KKT system

$$\begin{cases} u \geq 0, Ax - b \leq 0, u^T(Ax - b) = 0, \\ Qx + d + G^T v + A^T u = 0, \\ Gx - f = 0. \end{cases} \tag{4}$$

x^* is called a KKT point of (1)–(3) and a pair $(u^{*T}, v^{*T})^T$ is called the Lagrangian multiplier vector corresponding to x^* .

Theorem 2.5 [5] If Q is a positive definite matrix, then x^* is an optimal solution of (1)–(3), if and only if x^* is a KKT point of (1)–(3).

Lemma 2.6 [32] If A is an $n \times n$ nonsingular matrix, then the homogeneous system $AX = 0$ has only the trivial solution $X = 0$.

A Gradient Neurodynamic Model

We can establish the relationship between the solution to problem (1)–(3) and the solution to an equivalent unconstrained minimization problem via a merit function [13]. A merit function is a function whose global minimizers coincide with the solutions of the NCP. The class of NCP-functions defined below is used to construct a merit function.

Definition 3.1 • A function $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}$, is called an NCP-function if it satisfies

$$\phi(a, b) = 0 \iff a \geq 0, b \geq 0, ab = 0.$$

• A popular NCP-function is the FB function, which is strongly semismooth [30, 33] and is defined as

$$\phi_{\text{FB}}(a, b) = (a + b) - \sqrt{a^2 + b^2}.$$

The FB merit function $\psi_{\text{FB}} : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}_+$ can be obtained by taking the square of ϕ_{FB} , i.e.,

$$\psi_{\text{FB}}(a, b) = \frac{1}{2} \|\phi_{\text{FB}}(a, b)\|^2.$$

The perturbed FB function is also given by

$$\phi_{\text{FB}}^\varepsilon(a, b) = (a + b) - \sqrt{a^2 + b^2 + \varepsilon}, \quad \varepsilon \rightarrow 0_+.$$

The important property of $\phi_{\text{FB}}^\varepsilon$ can be stated in the following result.

Proposition 3.2 [10] *For every $\varepsilon \in \mathbb{R}$, we have*

$$\phi_{\text{FB}}^\varepsilon(a, b) = 0 \Leftrightarrow a > 0, b > 0, ab = \frac{\varepsilon}{2}.$$

The above proposition is obvious, and the proof is omitted. It is noted that $\phi_{\text{FB}}^\varepsilon(a, b)$ is smooth with respect to a, b for $\varepsilon > 0$.

Lemma 3.3 $y^* = (x^{*T}, u^{*T}, v^{*T})^T$ satisfies the following equation

$$\eta(y) = \begin{bmatrix} Qx + d + G^T v + A^T u \\ f - Gx \\ \phi_{\text{FB}}^\varepsilon(u, b - Ax) \end{bmatrix} = 0, \quad (5)$$

if and only if x^* is a KKT point of (1)–(3) for every $\varepsilon \rightarrow 0_+$.

Proof Let $\eta(y^*) = 0$. Then

$$Qx^* + d + G^T v^* + A^T u^*, \quad (6)$$

$$f - Gx^* = 0, \quad (7)$$

$$\phi_{\text{FB}}^\varepsilon(u^*, b - Ax^*) = 0, \quad \text{for } \varepsilon \rightarrow 0_+. \quad (8)$$

From proposition 3.2, it follows easily that $\phi_{\text{FB}}^\varepsilon(u^*, b - Ax^*) = 0$ if and only if

$$u^* \geq 0, \quad Ax^* - b \leq 0, \quad u^{*T}(Ax^* - b) = 0, \quad \text{for } \varepsilon \rightarrow 0_+. \quad (9)$$

From (6), (7) and (9), it is seen that $y^* = (x^{*T}, u^{*T}, v^{*T})^T$ satisfies the KKT conditions (4).

The converse is straightforward. □

Now by Definition 3.1 and Lemma 3.3, we can easily verify that the KKT system (4) is equivalent to the following unconstrained smooth minimization problem:

$$\text{minimize } E(y) = \frac{1}{2} \|\eta(y)\|^2. \quad (10)$$

It is clear that $E(y)$ is a smooth merit function for the KKT system (4). The merit function E in (10) is continuously differentiable for all $y \in \mathbb{R}^{n+l+m}$.

Now let $x(\cdot), u(\cdot)$ and $v(\cdot)$ be some time dependent variables. Our aim is to design a neural network that will settle down to an equilibrium point, which is also a stationary point of the energy function $E(y)$. Hence, we can use the steepest descent method to construct the following neural network model for solving problem (1)–(3) as

$$\frac{dy(t)}{dt} = -k \nabla E(y(t)), \quad k > 0, \quad (11)$$

$$y(0) = y_0, \quad (12)$$

where k is a scaling factor and indicates the convergence rate of the neural network (11) and (12). An indication on how the neural network (11) and (12) can be implemented on hardware is provided in Fig. 1.

In order to see how well the presented neural network (11) and (12) can be applied to solve SCQP problems, we compare it with some existing neural network models. First, let us consider the following problem

$$\text{minimize } \frac{1}{2} x^T A_0 x + a^T x \quad (13)$$

subject to

$$x \in \Omega_0, \quad (14)$$

$$Dx = b, \quad (15)$$

where $A_{0_{m \times m}}$ is a symmetric positive semidefinite matrix, D in an $n \times m$ matrix, $\Omega_0 \subset \mathbb{R}^m$ is a closed convex set and $x = (x_1, x_2, \dots, x_m)^T$. In [36], the projection neural network for solving (13)–(15) is given by

$$\frac{dx}{dt} = P_0(x - A_0 x + D^T y - a) - x, \quad (16)$$

$$\frac{du}{dt} = -DP_0(x - A_0 x + D^T y - a) + b, \quad (17)$$

where P_0 is the projection operator [40] on Ω_0 . The neural network model (16)–(17) has not been proved to be convergent in finite time. If we also use the network in [39] to solve the problem (13)–(15), the dimension of the network

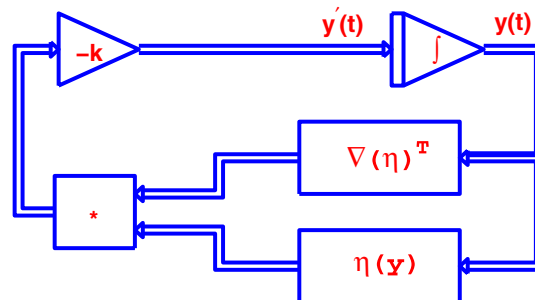


Fig. 1 A simplified block diagram for the neural network (11) and (12)

is higher. Furthermore, it has not been proved to be convergent in finite time, too.

There is another kind of the neural network model called gradient model. In order to use the gradient neural network model, a constrained optimization problem can be approximated by an unconstrained optimization problem. Then the energy function is constructed by the penalty function method. It is noticeable that the gradient neural network model has an advantage as the model may be defined directly using the derivatives of the energy function. But its shortcoming is that the convergence is not guaranteed, especially in the case of unbounded solution sets [46]. Moreover, the gradient neural network based on the penalty function requires any adjustable parameter called the penalty parameter. For instance, a gradient neural network model for solving (1)–(3) is given by

$$\frac{dx}{dt} = -(Qx + d + \gamma[A^T(Ax - b)^+ + G^T(Gx - f)]), \quad (18)$$

where γ is a penalty parameter. The system in (18) is referred to as Kennedy and Chua’s neural network model [17]. This network is not capable to find an exact optimal solution due to a finite penalty parameter and is difficult to implement when the penalty parameter is very large [18]. Thus, this network only converges to an approximate solution of (1)–(3) for any given finite penalty parameter. It can be also shown that the Kennedy–Chua’s neural network (18) is not globally convergent to an exact optimal solution of some quadratic programming problems. For instance, one can see Example 5.1 in this manuscript.

Stability and Convergence Properties

In order to study the stability of the neural network in (11) and (12), we first prove the following lemma.

Lemma 4.1 *The Jacobian matrix $\nabla\eta(y)$ of the mapping η defined in (5) is nonsingular.*

Proof With a simple calculation and using Proposition 3.2 in [35] and Proposition 3.3 in [34], it is clearly shown that

$$\nabla\eta(y) = \begin{bmatrix} Q & G^T & A^T \\ -G & O_{l \times l} & O_{l \times m} \\ (-\text{diag}\{\nabla_{(Ax-b)_k} \phi_{FB}^e(u, (b - Ax)_k)\}_{k=1}^m)A & O_{m \times l} & \text{diag}\{\nabla_{u_k} \phi_{FB}^e(u_k, (b - Ax)_k)\}_{k=1}^m \end{bmatrix}.$$

Since Q is a positive definite matrix and $\text{rank}(A, G) = m + l$, from Theorem 3.1 in [34] we see that $\nabla\eta(y)$ is nonsingular. This completes the proof. \square

We now investigate the relationships between an equilibrium point of (11) and (12) and a solution to problem (1)–(3).

Theorem 4.2 *Let x^* be a KKT point of (1)–(3). Then y^* is an equilibrium point of the neural network (11) and (12). On the other hand, if $y^* = (x^{*T}, u^{*T}, v^{*T})^T$ be an equilibrium point of the neural network (11) and (12) and the Jacobian matrix of $\eta(y)$ in (5) is nonsingular, then x^* is a KKT point of the problem (1)–(3).*

Proof Suppose that x^* is an optimal solution for (1)–(3). From Lemma 3.3, it is clear that $\eta(y^*) = 0$. With a simple calculation, it is clearly shown that

$$\nabla E(y) = (\nabla\eta(y))^T \eta(y), \quad (19)$$

where $\nabla\eta(y)$ is the Jacobian matrix of $\eta(y)$. Using (19) we get $\nabla E(y^*) = 0$, i.e. y^* is an equilibrium point of dynamic model (11) and (12). From Lemma 4.1 and Lemma 2.6, the converse of the proof is straightforward. \square

Lemma 4.3 *The equilibrium point of the proposed neural network model (11) and (12) is unique.*

Proof Since problem (1)–(3) has unique optimal solution x^* , the necessary and sufficient KKT conditions (4) has a unique solution $y^* = (x^{*T}, u^{*T}, v^{*T})^T$. Moreover, from Theorem 4.2 we see that the solution of the KKT system (4) is the equilibrium point of the proposed neural network model (11) and (12). Thus the equilibrium point of the network (11) and (12) is unique. \square

Now we state the main result of this section.

Theorem 4.4 *Let y^* be an isolated equilibrium point of (11) and (12). Then y^* is asymptotically stable for (11) and (12).*

Proof First, notice that $E(y) \geq 0$ and $E(y^*) = 0$. In addition, since y^* is an isolated equilibrium point of (11) and (12), there exists a neighborhood $\Omega_* \subseteq \mathbb{R}^{n+m+l}$ of y^* such that $\nabla E(y^*) = 0$, and $\nabla E(y) \neq 0, \forall y \in \Omega_* \setminus \{y^*\}$.

We claim that for any $y \in \Omega_* \setminus \{y^*\}$, $E(y) > 0$. Otherwise if there is a $y \in \Omega_* \setminus \{y^*\}$ satisfying $E(y) = 0$. Then, by (10)

and (19) we have $\nabla E(y) = 0$, i.e., y is also an equilibrium point of (11), which clearly contradicts the assumption that y^* is an isolated equilibrium point in Ω_* . Moreover

$$\frac{dE(y(t))}{dt} = [\nabla E(y(t))]^T \frac{dy(t)}{dt} = -k \|\nabla E(y(t))\|^2 \leq 0, \tag{20}$$

and

$$\frac{dE(y(t))}{dt} < 0, \quad \forall y(t) \in \Omega_* \text{ and } y(t) \neq y^*.$$

This by lemma 2.2 (b) implies that y^* is asymptotically stable. \square

Theorem 4.5 *Suppose that $y = y(t, y_0)$ is a trajectory of (11) and (12) in which the initial point is $y_0 = y(0, y_0)$ and the level set $L(y_0) = \{y \in \mathbb{R}^{n+m+l} : E(y) \leq E(y_0)\}$ is bounded, then*

- (a) $\gamma^+(y_0) = \{y(t, y_0) | t \geq 0\}$ is bounded.
- (b) There exists \bar{y} such that $\lim_{t \rightarrow \infty} y(t, y_0) = \bar{y}$.

Proof (a) Suppose y^* is an equilibrium point of the network in (11) and (12). Calculating the derivative of $E(y)$ along the trajectory $y(t, y_0)$, ($t \geq 0$) one has

$$\frac{dE(y(t))}{dt} = [\nabla E(y(t))]^T \frac{dy}{dt} = -k \|\nabla E(y(t))\|^2 \leq 0. \tag{21}$$

Thus $E(y)$ is monotone nonincreasing along the trajectory $y = y(t, y_0)$, ($t \geq 0$). Therefore $\gamma^+(y_0) \subseteq L(y_0)$, that is to say $\gamma^+(y_0) = \{y(t, y_0) | t \geq 0\}$ is bounded.

(b) By (a) $\gamma^+(y_0) = \{y(t, y_0) | t \geq 0\}$ is a bounded set of points. Take strictly monotone increasing sequence $\{\bar{t}_n\}$, $0 \leq \bar{t}_1 \leq \dots \leq \bar{t}_n \leq \dots$, $\bar{t}_n \rightarrow +\infty$, then $\{y(\bar{t}_n, y_0)\}$ is a bounded sequence composed of infinitely many points. Thus there exists limiting point \bar{y} , that is, there exists a subsequence $\{t_n\} \subseteq \{\bar{t}_n\}$, $t_n \rightarrow +\infty$ such that

$$\lim_{n \rightarrow +\infty} y(t_n, y_0) = \bar{y},$$

where \bar{y} satisfies

$$\frac{dE(y(t))}{dt} = 0,$$

which indicates that \bar{y} is ω -limit point of $\gamma^+(y_0)$. Using the LaSalle Invariant Set Theorem [12], one has that $y(t, y_0) \rightarrow \bar{y} \in M$ as $t \rightarrow \infty$, where M is the largest invariant set in $K = \{y(t, y_0) | \frac{dE(y(t, y_0))}{dt} = 0\}$. From (11), (12) and (21), it follows that $\frac{dx}{dt} = 0$, $\frac{du}{dt} = 0$ and $\frac{dv}{dt} = 0 \Leftrightarrow \frac{dE(y(t))}{dt} = 0$; thus $\bar{y} \in D^*$ by $M \subseteq K \subseteq D^*$. Therefore, from any initial state y_0 , the trajectory $y(t, y_0)$ of (11) and (12) tends to \bar{y} . The proof is complete. \square

As an immediate corollary of Theorems 4.4 and 4.5, we can get the following result.

Corollary 4.6 *If $D^* = \{(x^{*T}, u^{*T}, v^{*T})^T\}$, then the neural network (11) and (12) for solving (1)–(3) is globally asymptotically stable to the unique equilibrium point $y^* = (x^{*T}, u^{*T}, v^{*T})^T$, where D^* is denoted as the optimal point set of (1)–(3).*

Computer Simulations

In order to demonstrate the effectiveness of the proposed neural network (11) and (12), in this section we test several examples. For each test problem, we compare the numerical performance of the proposed neural network with various values of k and various initial states $y(0)$. We also provide two applicable examples in engineering as support vector machine learning in regression and constrained least-squares approximation problem. The simulation is conducted on Matlab 7, the ODE solver engaged is ode45s.

Example 5.1

$$\text{minimize } f(x) = x_1^2 + x_2^2 + x_1x_2 - 30x_1 - 30x_2$$

$$\text{subject to } \begin{cases} \frac{5}{12}x_1 - x_2 \leq \frac{35}{12}, \\ \frac{5}{2}x_1 + x_2 \leq \frac{35}{2}, \\ -x_1 \leq 5, \\ x_2 \leq 5. \end{cases}$$

The problem and its dual have a unique solution $(x^{*T}, u^{*T})^T = (5, 5, 0, 6, 0, 9)^T$. Theorems 4.4 and 4.5 and corollary 4.6 guarantee that the stated model in (11) and (12) converges globally to $x^* = (5, 5)^T$. Figure 2 depicts the phase diagram of state variables $(x_1(t), x_2(t))^T$, with 5 different initial points and scaling factor $k = 10$, where S stands for the feasible region. These vectors converge to its exact solution x^* globally.

We test the influence of the parameter k on the value of $\|y(t) - y^*\|^2$. From Fig. 3, we see that when $k = 0.1$, the neural network (11) and (12) generates the slowest decrease of $\|y(t) - y^*\|^2$, whereas when $k = 10$, it generates the fastest decrease of $\|y(t) - y^*\|^2$ with the initial state $y_0 = (2, -2, 2, -2, 2, -2)^T$. Figure 4 also describes the convergence behavior of $\|y(t) - y^*\|^2$ with 10 various initial states and $k = 5$.

To make a comparison, the above problem is solved using the Kennedy and Chua’s neural network in (18) with $\gamma = 50$ in [22]. It is seen that this network converges to its equilibrium point $\bar{x} = (4.97778, 5.1745)^T$, which can be viewed as the approximate solution of the above problem. It is clear that this equilibrium point is not feasible. Thus, we can conclude that the proposed network (11) and (12) is feasible and has a good stability performance.

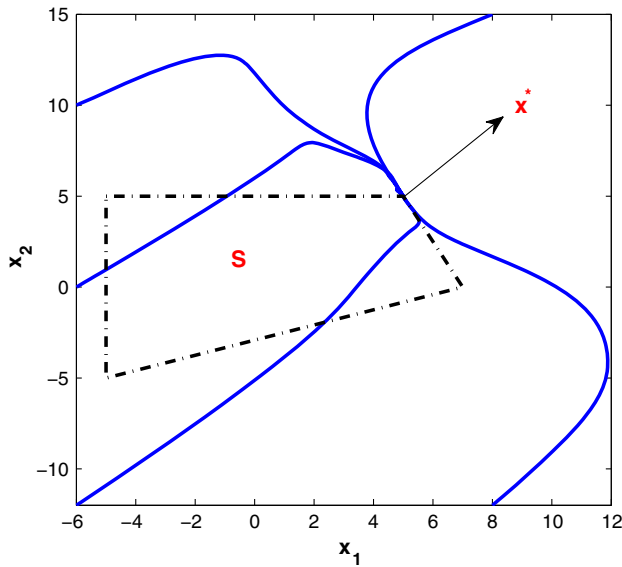


Fig. 2 Transient behavior of the neural network (11) and (12) with five different initial points and $k = 10$ in Example 5.1

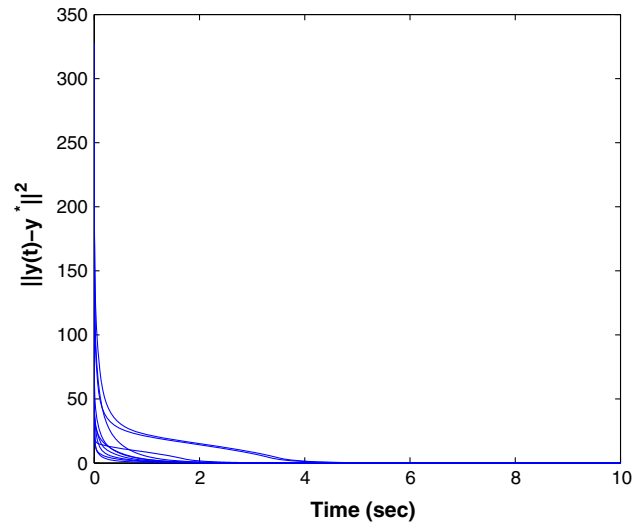


Fig. 4 Convergence behavior of $\|y(t) - y^*\|^2$ with 10 different initial points and $k = 5$ in Example 5.1

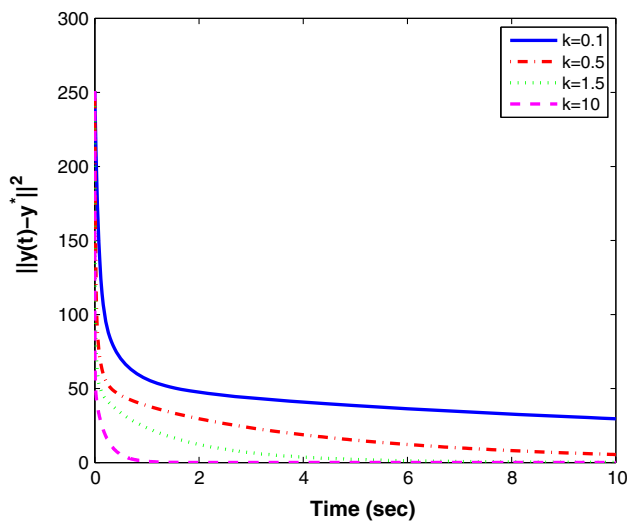


Fig. 3 Convergence behavior of $\|y(t) - y^*\|^2$ with $y_0 = (2, -2, 2, -2, 2, -2)^T$ in Example 5.1

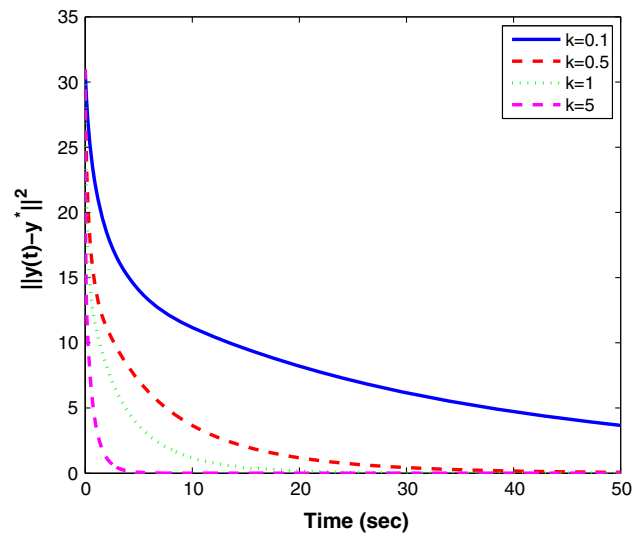


Fig. 5 Convergence behavior of $\|y(t) - y^*\|^2$ with $y_0 = (1, -1, 1, -1, 1, -1, 1, -1)^T$ in Example 5.2

Example 5.2

$$\text{minimize } f(x) = x_1^2 + x_2^2 + x_3^2$$

$$\text{subject to } \begin{cases} 2x_1 + x_2 - 5 \leq 0, \\ x_1 + x_3 - 2 \leq 0, \\ -x_1 + 1 \leq 0, \\ -x_2 + 2 \leq 0, \\ -x_3 \leq 0. \end{cases}$$

The optimal solution is $y^* = (1, 2, 0, 0, 0, 2, 4, 0)^T$. Figure 5 shows the local convergence behavior of the error $\|y(t) - y^*\|^2$ with different k and the initial point

$y_0 = (1, -1, 1, -1, 1, -1, 1, -1)^T$. It is clear that a larger k yields a better convergence rate of the error $\|y(t) - y^*\|^2$. Figure 6 shows that the trajectories of the proposed neural network in (11) and (12) to solve the problem with 6 random initial points and $k = 5$ converge to the optimal solution of this problem. It is easy to verify that whether or not an initial point is taken inside or outside the feasible region, the proposed network always converges to the theoretical optimal solution $x^* = (1, 2, 0)^T$.

Example 5.3

$$\text{minimize } f(x) = x_1^2 + x_2^2 + x_3^2 - 2x_1 - 3x_4$$

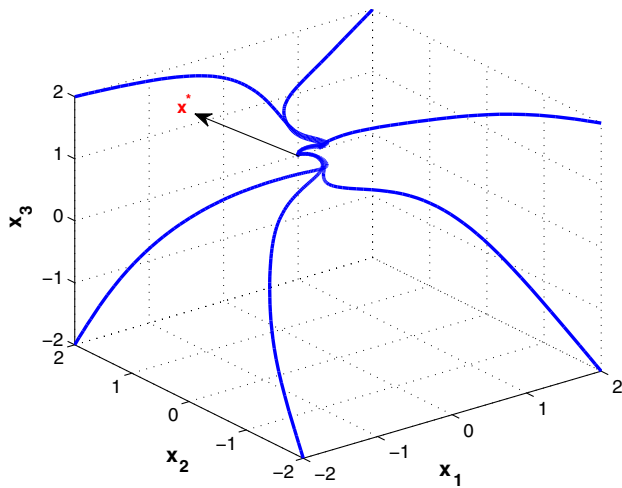


Fig. 6 Transient behavior of $x(t)$ with six different initial points and $k = 5$ in Example 5.2

$$\text{subject to } \begin{cases} -x_1 \leq 0, \\ -x_2 \leq 0, \\ -x_3 \leq 0, \\ -x_4 \leq 0, \\ 2x_1 + x_2 + x_3 + 4x_4 = 7, \\ x_1 + x_2 + 2x_3 + x_4 = 6. \end{cases}$$

The optimal solution for this example is $x^* = (1.1233, 0.6507, 1.8288, 0.5685)^T$, $u^* = (0, 0, 0, 0)^T$ and $v^* = (1.0548, -2.3562)^T$. Figures 7 and 8 display the transient behavior of the proposed network with 10 different initial points and $k = 1$. All trajectories of the network converge to the optimal solution $y^* = (x^{*T}, u^{*T} - v^{*T})^T$. Moreover, when the initial point is chosen as an infeasible point, the trajectory of the network still converges to y^* . Figure 9 shows how the value of $\|y(t) - y^*\|^2$ with 10 various initial states and $k = 1$.

Example 5.4 Consider the regression problem of approximating a set of data

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}, \tag{22}$$

with a regression function as

$$\phi(x) = \sum_{i=1}^N \alpha_i \Phi_i(x) + \beta, \tag{23}$$

where $\Phi_i(x) (i = 1, \dots, N)$ are called feature functions defined in a high-dimensional space and $\alpha_i (i = 1, \dots, N)$ and β are parameters of the model to be estimated. Here, we can use the recurrent neural network in (11) and (12) to estimate these parameters. By utilizing Huber loss function [41], the regression function defined in (23) can be represented as

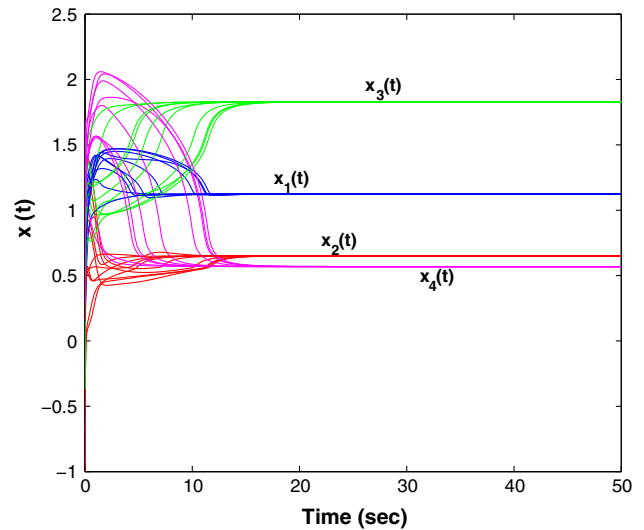


Fig. 7 Transient behavior $x(t)$ of the proposed neural network with 10 different initial points and $k = 1$ in Example 5.3

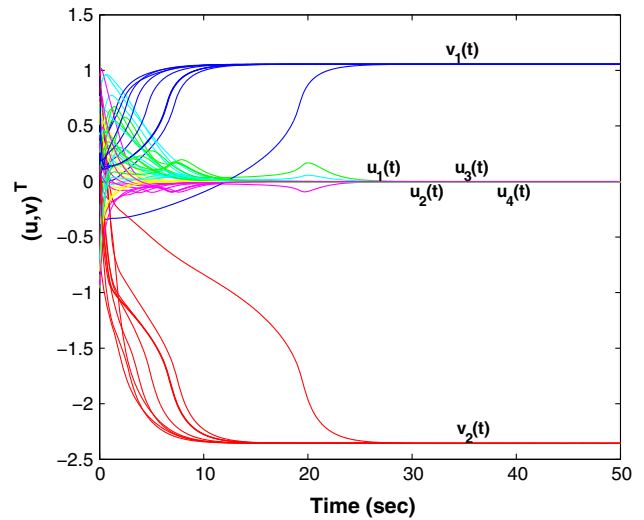


Fig. 8 Transient behavior $(u(t)^T, v(t)^T)^T$ of the proposed neural network with 10 different initial points and $k = 1$ in Example 5.3

$$\phi(x) = \sum_{i=1}^N \xi_i K(x, x_i) + \beta,$$

where $K(x, y)$ is a kernel function satisfying $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$. According to the problem formulation in [20], $\xi_i (i = 1, \dots, N)$ is the optimal solution of the following quadratic programming problem:

$$\text{minimize } \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \xi_i \xi_j K(x_i, x_j) - \sum_{i=1}^N \xi_i y_i + \frac{\zeta}{2Q} \sum_{i=1}^N \xi_i^2 \tag{24}$$

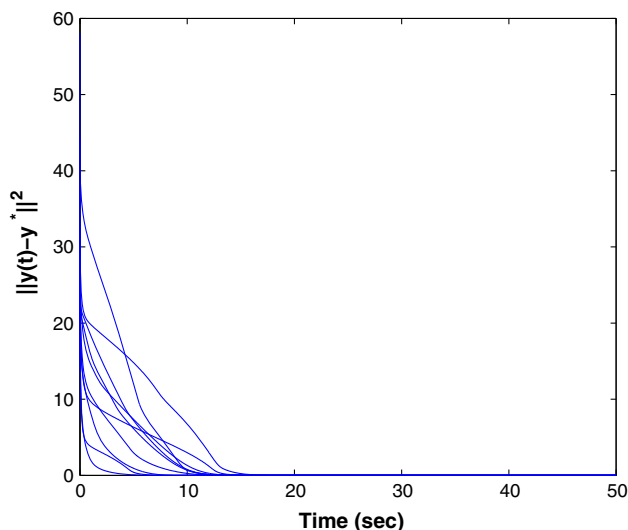


Fig. 9 Convergence behavior of $\|y(t) - y^*\|^2$ with 10 different initial points and $k = 1$ in Example 5.3

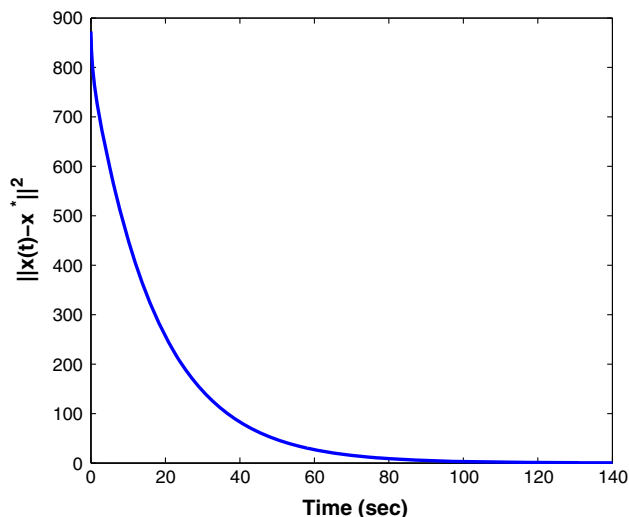


Fig. 11 Convergence behavior of $\|x(t) - x^*\|^2$ with $x_0 = (1, -1, 1, -1, 1, -1, 1, -1, 1)^T$ and $k = 10$ in Example 5.4

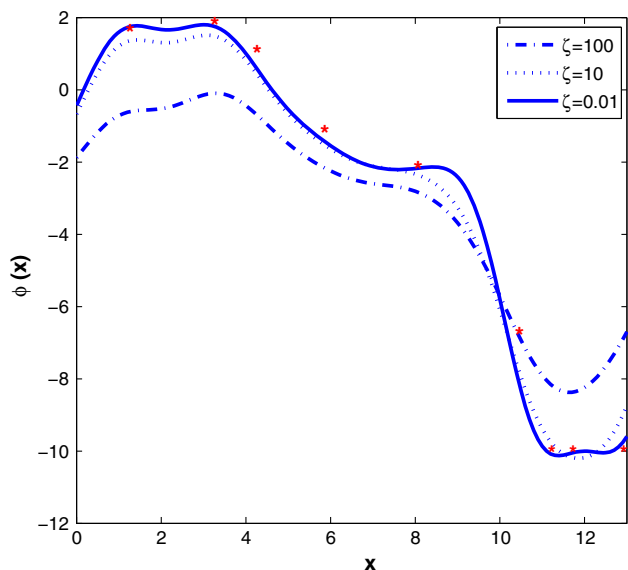


Fig. 10 Results of support vector regression using the proposed neural network (11) and (12) with an RBF kernel where $\varrho = 100$, $\delta = 1$ and three different ζ parameters in Example 5.4

$$\text{subject to } \begin{cases} \sum_{i=1}^N \zeta_i = 0, \\ -\varrho \leq \zeta_i \leq \varrho, \quad i = 1, \dots, N, \end{cases} \quad (25)$$

where $\zeta > 0$ is an accuracy parameter required for the approximation and $\varrho > 0$ is a prespecified parameter. It is also shown [3] that $\beta = v^*$, where v^* is the equilibrium point of the proposed neural network model to solve the above quadratic optimization problem.

As an example, we consider the regression data in Table 1. We use the proposed neural network (11) and (12) with an RBF kernel to train an support vector machine for the regression problem. We choose the following Gaussian function [21]

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\delta^2}\right).$$

Figure 10 depicts the regression results with $\varrho = 100$, $\delta = 1$ and three different ζ parameters. It is clear that when ζ tends to zero, the approximation is more accurate. Figure 11 also shows the convergence of the error norm between the output vector and the optimal solution x^* with the initial point $x_0 = (1, -1, 1, -1, 1, -1, 1)^T$. The regression results illustrate the good performance of the proposed neural network.

Example 5.5 Let us consider a constrained least-squares approximation problem [19]: Find the parameters of the combination of exponential and polynomial functions $y(x) = a_4e^x + a_3x^3 + a_2x^2 + a_1x + a_0$, which fit the data given in Table 2 and subjects to the constraints $8.1 \leq y(1.3) \leq 8.3$, $3.4 \leq y(2.8) \leq 3.5$ and $2.25 \leq y(4.2) \leq 2.26$. This problem can be formulated as follows:

$$\text{minimize } \|Cx - d\|^2$$

subject to

$$Ax \in \Omega,$$

$$\text{where } x = (x_1, x_2, x_3, x_4, x_5)^T = (a_4, a_3, a_2, a_1, a_0)^T,$$

Table 1 Regression data for Example 5.4

x	1.0	3.0	4.0	5.6	7.8	10.2	11.0	11.5	12.7
y	1.6	1.8	1.0	-1.2	-2.2	-6.8	-10.0	-10.0	-10.0

Table 2 Approximation data for Example 5.5

x	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5
y	7.6	7.2	7.9	8	6.2	6.2	3	0.8	1.2	5.8

$$C = \begin{bmatrix} 1 & 1.649 & 2.718 & 4.482 & 7.389 & 12.183 & 20.086 & 33.116 & 54.598 & 90.017 \\ 0 & 0.125 & 1 & 3.375 & 8 & 15.625 & 27 & 42.875 & 64 & 91.125 \\ 0 & 0.25 & 1 & 2.25 & 4 & 6.25 & 9 & 12.25 & 16 & 20.25 \\ 0 & 0.5 & 1 & 1.5 & 2 & 2.5 & 3 & 3.5 & 4 & 4.5 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T,$$

$$d = [7.6 \ 7.2 \ 7.9 \ 8 \ 6.2 \ 6.2 \ 3 \ 0.8 \ 1.2 \ 5.8]^T,$$

$$A = \begin{bmatrix} 3.669 & 2.197 & 1.69 & 1.3 & 1 \\ 16.445 & 21.952 & 7.84 & 2.8 & 1 \\ 66.686 & 74.088 & 17.64 & 4.2 & 1 \end{bmatrix},$$

and $\Omega = \{x \in \mathbb{R}^3 : 8.1 \leq x_1 \leq 8.3, 3.4 \leq x_2 \leq 3.5, 2.25 \leq x_3 \leq 2.26\}$. Figure 12 depicts the approximation results using the proposed neural network. An l_2 norm error between x and x^* with the initial point $x_0 = (2, -2, 2, -2, 2)^T$ is also shown in Fig. 13.

Example 5.6 We apply the proposed artificial neural network (11) and (12) to increase the useful information content of images and improve the quality of the noisy images. Consider an array of n sensors. Let $I_l(k)$ denotes

the received $M \times N$ two dimensional gray-level image from the l 'th sensor, whose amplitude is denoted by $f_l(i, j)$. We assume

$$I_l((i - 1)N + j) = f_l(i, j), \quad i = 1, \dots, M; j = 1, \dots, N.$$

Then, the n - dimensional vector of information received from n sensors is given by

$$I(k) = as(k) + \hat{n}(k),$$

where $a = [a_1, \dots, a_n]^T$, $I_k = [I_1(k), \dots, I_n(k)]^T$ and $\hat{n}(k) = [\hat{n}_1(k), \dots, \hat{n}_n(k)]^T$, a_l is a scaling coefficient, $s(k)$ denotes the signal and $\hat{n}_l(k)$ represents the additive Gaussian noise at l 'th sensor with zero mean. Moreover, $s(k)$ and $\hat{n}_l(k)$ are mutually independent random processes. According to the result discussed in [23], the main

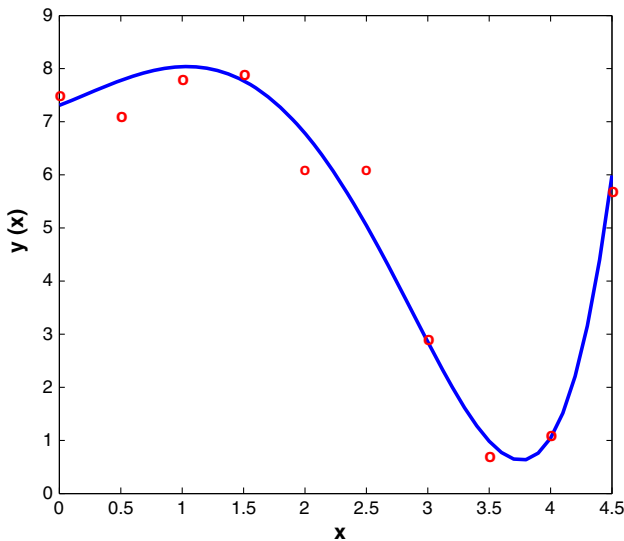


Fig. 12 Approximation results of the neural network (11) and (12) in Example 5.5, where the circles represent the approximation data and the curve is the approximation function

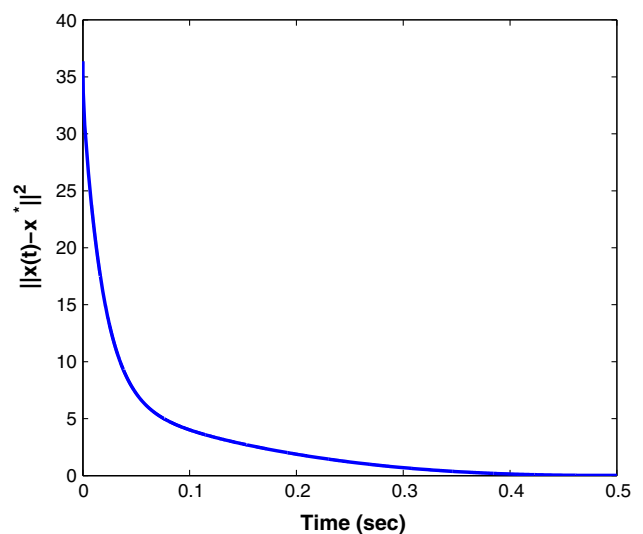


Fig. 13 Convergence behavior of $\|x(t) - x^*\|^2$ with $x_0 = (2, -2, 2, -2, 2)^T$ and $k = 7$ in Example 5.5

contrivance for image fusion that we address here is to seek an optimal fusion such that the uncertainty of the fused information is minimized. The optimal fused information is given by $s^*(k) = \sum_{l=1}^n x_l^* I_l(k)$, where the optimal fusion x^* is chosen according to the following deterministic quadratic programming problem:

$$\text{minimize } x^T R x \quad (26)$$

$$\text{subject to } \begin{cases} a^T x = 1, \\ x \geq 0, \end{cases} \quad (27)$$

where $a = [1, \dots, 1]^T \in \mathbb{R}^n$ and $R = \frac{1}{MN} \sum_{k=1}^{MN} I(k) I(k)^T$. The artificial neural network (11) and (12) converges to the optimal fusion vector x^* . In order to see the fused image we convert s^* to f^* as

$$f^*(i, j) = s^*((i-1)N + j), \quad i = 1, \dots, M; j = 1, \dots, N.$$

Finally, we use the function “`imshow(f^*)`” in Matlab software. One can improve the quality of the fused images using the proposed neural image fusion algorithm by increasing the number of sensors.

As an example, we illustrate the performance of the proposed neural image fusion algorithm for gray-level Lena images, as shown in Fig. 14. These Lane images are eight bit gray-level images with 206×245 pixels. Figure 14a is the display of a typical noisy measurement of the Lane image at one sensor, where SNR is 10 dB. Figure 14b–d show the fused images by the proposed algorithm for the number of sensors $n = 10, 20$ and 30 , respectively. It is easy to verify that the quality of images is improving when the number of sensors increases.

To end this section, we answer a natural question: are there advantages of the proposed neural network compared

Fig. 14 Lena image fusion using neural image fusion algorithm. **a** The noisy image. **b–d** The fused image with $n = 10, 20$ and 30 sensors in Example 5.6



to the existing ones? To answer this, we summarize what we have observed from numerical experiments and theoretical results as below.

- Compared with traditional numerical optimization algorithms, the neural network approach has several potential advantages in real-time applications. First, the structure of a neural network can be implemented effectively using very large scale integration and optical technologies. Second, neural networks can solve many optimization problems with time-varying parameters. Third, the dynamical techniques and the numerical ODE techniques can be applied directly to the continuous-time neural network for solving constrained optimization problems effectively.
- We compare our neural network model with some existing models which also work for quadratic optimization problems, for instance, the ones used in (16), (17) and (18). At first glance, these neural network models look having lower complexity. However, we observe that the difference in the numerical performance is very marginal by testing some quadratic optimization problems.
- Changing initial points may not have much effect for our neural network model, whereas it does for some existing models. The reason is that our model is globally convergent to the optimal solution of the problem.
- Three examples with respect to support vector machines for regression, constrained least-squares approximation problem and image fusion algorithm for gray-level images are presented to indicate the real-time applications of the proposed neural network.

Conclusion

In this paper, we have proposed a new neural network for solving SCQP problems. Based on the duality theory of convex programming, FB function, KKT optimality conditions, convex analysis theory, Lyapunov stability theory and LaSalle invariance principle, the constructed network can find the optimal solution of the primal and dual problems simultaneously. Compared with the gradient-based networks available, the structure of the proposed network is reliable and efficient. The other advantages of the proposed neural network are that it can be implemented without a penalty parameter and can be convergent to an exact solution to SCQP program with general linear constraints. Moreover, the proposed transformation method in this article can allow us to transform easily and efficiently general constrained SCQP programming problems into unconstrained problems. In this article, we also analyze the

influence of the parameter k on the convergence rate of the trajectory and the convergence behavior of $\|y(t) - y^*\|^2$ and obtain that a larger k leads to a better convergence rate. The simulation results have demonstrated the global convergence behaviors and characteristics of the proposed neural network for solving several SCQP problems.

References

1. Agrawal SK, Fabien BC. Optimization of dynamic systems. Netherlands: Kluwer Academic Publishers; 1999.
2. Ai W, Song YJ, Chen YP. An improved neural network for solving optimization of quadratic programming problems. In: Proceedings of the fifth international conference on machine learning and cybernetics, Dalian; 2006. p.13–6.
3. Anguita D, Boni A. Improved neural network for SVM learning. IEEE Trans Neural Netw. 2002;13:1243–44.
4. Avriel M. Nonlinear programming: analysis and methods. Englewood Cliffs, NJ: Prentice-Hall; 1976.
5. Bazaraa MS, Sherali HD, Shetty CM. Nonlinear programming—theory and algorithms, 2nd ed. New York: Wiley; 1993.
6. Bertsekas DP. Parallel and distributed computation: numerical methods. Englewood Cliffs, NJ: Prentice-Hall; 1989.
7. Boggs PT, Domic PD, Rogers JE. An interior point method for general large-scale quadratic programming problems. Ann Oper Res. 1996;62:419–37.
8. Boland NL. A dual-active-set algorithm for positive semi-definite quadratic programming. Math Program. 1996;78:1–27.
9. Boyd S, Vandenberghe L. Convex optimization. Cambridge: Cambridge University Press; 2004.
10. Facchinei F, Jiang H, Qi L. A smoothing method for mathematical programs with equilibrium constraints. Math Program. 1999;35:107–34.
11. Fletcher R. Practical methods of optimization. New York: Wiley; 1981.
12. Hale JK. Ordinary differential equations. New York: Wiley-Interscience; 1969.
13. Hu SL, Huang ZH, Chen JS. Properties of a family of generalized NCP-functions and a derivative free algorithm for complementarity problems. J Comput Appl Math. 2009;230:69–82.
14. Huang YC. A novel method to handle inequality constraints for convex programming neural network. Neural Process Lett. 2002;16:17–27.
15. Jiang M, Zhao Y, Shen Y. A modified neural network for linear variational inequalities and quadratic optimization problems. Lecture Notes in Computer Science, 5553, Springer, Berlin: Heidelberg; 2009.p. 1–9.
16. Kalouptisidis N. Signal processing systems, theory and design. New York: Wiley; 1997.
17. Kennedy MP, Chua LO. Neural networks for nonlinear programming. IEEE Trans Circuits Syst. 1988;35:554–62.
18. Lillo WE, Loh MH, Hui S, Zäk SH. On solving constrained optimization problems with neural networks: a penalty method approach. IEEE Trans Neural Netw. 1993;4(6):931–39.
19. Liu Q, Cao J. Global exponential stability of discrete-time recurrent neural network for solving quadratic programming problems subject to linear constraints. Neurocomputing. 2011;74:3494–01.
20. Liu Q, Wang J. A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming. IEEE Trans Neural Netw. 2008;19:558–70.
21. Liu Q, Zhao Y. A continuous-time recurrent neural network for real-time support vector regression. In: Computational

- Intelligence in Control and Automation (CICA), 2013 IEEE Symposium on, 16–19 April 2013; p. 189–193.
22. Maa CY, Shanblatt MA. Linear and quadratic programming neural network analysis. *IEEE Trans Neural Netw.* 1992;3(4): 580–94.
 23. Malek A, Yashtini M. Image fusion algorithms for color and gray level images based on LCLS method and novel artificial neural network. *Neurocomputing.* 2010;73:937–43.
 24. Miller RK, Michel AN. *Ordinary differential equations.* New-York: Academic Press; 1982.
 25. More JJ, Toraldo G. On the solution of large quadratic programming problems with bound constraints. *SIAM J Optim.* 1991;1:93–13.
 26. Nazemi AR. A dynamical model for solving degenerate quadratic minimax problems with constraints. *J Comput Appl Math.* 2011; 236:1282–95.
 27. Nazemi AR. A dynamic system model for solving convex nonlinear optimization problems. *Commun Nonlinear Sci Numer Simul.* 2012;17:1696–05.
 28. Nazemi AR, Omid F. A capable neural network model for solving the maximum flow problem. *J Comput Appl Math.* 2012; 236:3498–13.
 29. Nazemi AR, Omid F. An efficient dynamic model for solving the shortest path problem. *Transp Res Part C Emerg Technol.* 2013; 26:1–19.
 30. Pan S-H, Chen J-S. A semismooth Newton method for the SOCCP based on a one-parametric class of SOC complementarity functions. *Comput Optim Appl.* 2010;45:59–88.
 31. Pyne IB. Linear programming on a electronic analogue computer. *Trans Am Inst Elect Eng.* 1956;75:139–43.
 32. Quarteroni A, Sacco R, Saleri F. *Numerical mathematics. Texts in applied mathematics, vol. 37, 2nd ed.* Berlin: Springer; 2007.
 33. Sun D, Sun J. Strong semismoothness of the Fischer–Burmeister SDC and SOC complementarity functions. *Math Program.* 2005; 103(3):575–81.
 34. Sun J, Zhang L. A globally convergent method based on Fischer–Burmeister operators for solving second-order cone constrained variational inequality problems. *Comput Math Appl.* 2009;58: 1936–46.
 35. Sun J, Chen J-S, Ko C-H. Neural networks for solving second-order cone constrained variational inequality problem. *Comput Optim Appl.* 2012;51:623–48.
 36. Tao Q, Cao J, Sun D. A simple and high performance neural network for quadratic programming problems. *Appl Math Comput.* 2001;124:251–60.
 37. Xia Y, Feng G. An improved network for convex quadratic optimization with application to real-time beamforming. *Neurocomputing.* 2005;64:359–74.
 38. Xia Y, Wang J. Primal neural networks for solving convex quadratic programs. In: *International joint conference on neural networks*; 1999. p. 582–87.
 39. Xia Y, Wang J. A recurrent neural network for solving linear projection equations. *Neural Netw.* 2000;13:337–50.
 40. Xia Y, Wang J. A general projection neural network for solving monotone variational inequality and related optimization problems. *IEEE Trans Neural Netw.* 2004;15:318–28.
 41. Xia Y, Wang J. A one-layer recurrent neural network for support vector machine learning. *IEEE Trans Syst Man Cybern Part B: Cybern.* 2004;34:1261–69.
 42. Xia Y, Leung H, Wang J. A projection neural network and its application to constrained optimization problems. *IEEE Trans Circuits Syst.* 2002;49:447–58.
 43. Xia Y, Feng G. Solving convex quadratic programming problems by an modified neural network with exponential convergence. *IEEE International Conference Neural Networks and Signal Processing, Nanjing, China*; 2003. p. 14–17.
 44. Xia Y, Feng G, Wang J. A recurrent neural network with exponential convergence for solving convex quadratic program and related linear piecewise equation. *Neural Networks.* 2004;17: 1003–15.
 45. Xue X, Bian W. A project neural network for solving degenerate convex quadratic program. *Neurocomputing.* 2007;70:2449–59.
 46. Yang Y, Cao J. A feedback neural network for solving convex constraint optimization problems. *Appl Math Comput.* 2008;201: 340–50.
 47. Zhang J, Zhang L. An augmented lagrangian method for a class of inverse quadratic programming problems. *Appl Math Optim.* 2010;61:57–83.